Structured Programming Language (CSE113)
Lecture #06
Functions in C Programming

Function: A function is a block of code that performs a specific task.

Types of functions in C programming

There are two types of functions in C programming:

- Standard library functions
- User defined functions

Standard library functions

The standard library functions are built-in functions in C programming to handle tasks such as mathematical computations, I/O processing, string handling etc. These functions are defined in the header file. When you include the header file, these functions are available for use.

Example: printf(), scanf(), getchar(), fprintf() etc.

User-defined functions (UDF)

The functions are declared and defined by the programmer/user known as User Define Function.

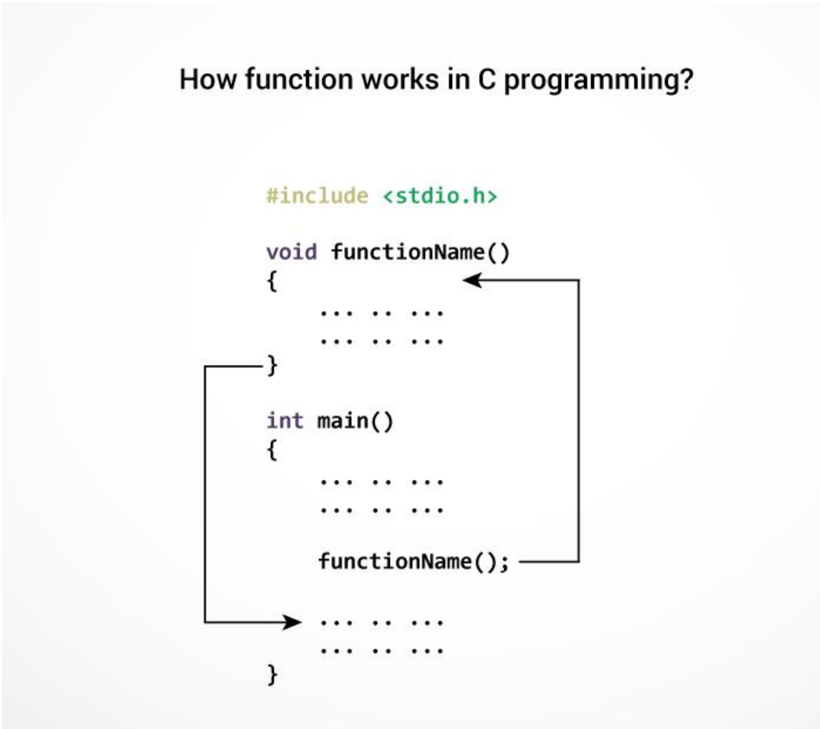
Example: SU(), add(), area() etc.

Advantages of user-defined function

- The program will be easier to understand, maintain and debug.
- Reusable codes that can be used in other programs
- A large program can be divided into smaller modules. Hence, a large project can be divided among many programmers.

Uses of C functions:

- Used to avoid rewriting same logic/code again and again in a program.
- There is no limit in calling C functions to make use of same functionality wherever required.
- We can call functions any number of times in a program and from any place in a program.
- A large C program can easily be tracked when it is divided into functions.
- The core concept of C functions are re-usability, dividing a big task into small pieces to achieve the functionality and to improve understandability of very large C programs.



C function declaration, function call and function definition:

There are 3 aspects in each C function. They are,

- **Function declaration or prototype** – This informs compiler about the function name, function parameters and return value's data type.
- **Function call** – This calls the actual function
- **Function definition** – This contains all the statements to be executed.

C functions aspects	syntax
function declaration	**return_type** function_name (**argument list**);
function definition	**Return_type** function_name (**arguments list**) { Body of function; }
function call	**function_name** (arguments list);

C function arguments and return values

All C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function. Now, we will see simple example C programs for each one of the below.

1. C function without arguments and without return value
2. C function with arguments and with return value
3. C function with arguments and without return value
4. C function without arguments and with return value

1. C function without arguments and without return value

Example: Add Two Numbers

Source Code:

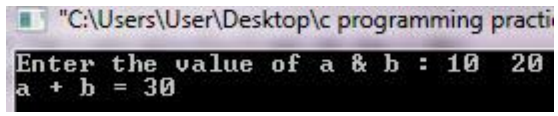
```
#include<stdio.h>

void add(); //function declaration

int main()
{
     add();  // function call
   return 0;
}

//function definition
void add()
{
   int a,b,sum;
   printf("Enter the value of a & b : ");
   scanf("%d %d",&a,&b);
   sum = a + b;
   printf("a + b = %d\n",sum);
}
```

output:



2. C function with arguments and with return value

Example: Add Two Numbers

Source Code:

```
#include<stdio.h>
int add(int x, int y); //function declaration
int main()
{
   int a,b,k;
   printf("Enter the value of a & b : ");
   scanf("%d %d",&a,&b);
     k = add(a,b);  // function call
   printf("a + b = %d\n",k);
   return 0;
}
//function definition
int add(int x, int y)
{
   int sum;
   sum = x + y;
   return sum;
}
```

output:



3. C function with arguments and without return value

Example: Add Two Numbers

Source Code:

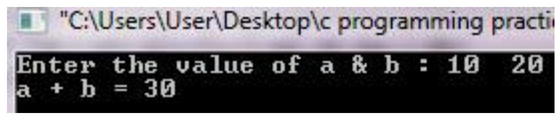
```
#include<stdio.h>

void add(int x, int y); //function declaration

int main()
{
   int a,b;
   printf("Enter the value of a & b : ");
   scanf("%d %d",&a,&b);
     add(a,b);  // function call
   return 0;
}

//function definition
void add(int x, int y)
{
   int sum;
   sum = x + y;
   printf("a + b = %d\n",sum);
}
```

output:



4. C function without arguments and with return value

Example: Add Two Numbers

Source Code:

```
#include<stdio.h>
int add(); //function declaration
int main()
{
   int k;
     k = add();  // function call
   printf("a + b = %d\n",k);
   return 0;
}
//function definition
int add()
{
   int a,b, sum;
   printf("Enter the value of a & b : ");
   scanf("%d %d",&a,&b);
       sum = a + b;
   return sum;
}
```

output:



5. Program to Swap Numbers Using Temporary Variable

Source Code:

```
#include<stdio.h>

void swap(); //function declaration

int main()
{
   swap(); //function call
   return 0;
}

//function definition
void swap(){
   int x, y, temp;

   printf("Enter the value of x : ");
   scanf("%d", &x);

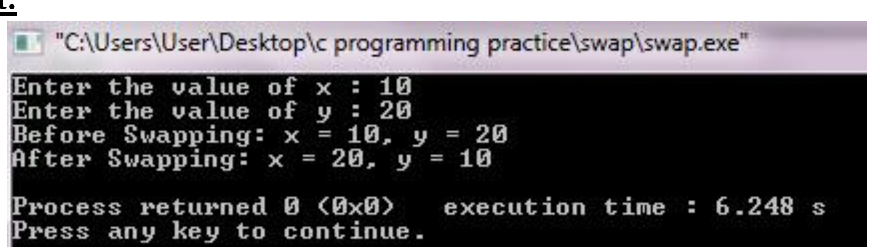
   printf("Enter the value of y : ");
   scanf("%d", &y);

   printf("Before Swapping: x = %d, y = %d\n", x, y);

      temp = x;
      x = y;
      y = temp;

   printf("After Swapping: x = %d, y = %d\n", x, y);
}
```

Output:



6. Program to Swap Numbers without Using Temporary Variable

a) **using + and – operators**
b) **using * and / operators**
c) **using bitwise X-OR**

a) <u>Source Code:</u>

```
#include<stdio.h>

void swap();//function declaration

int main()
{
   swap();//function call
   return 0;
}

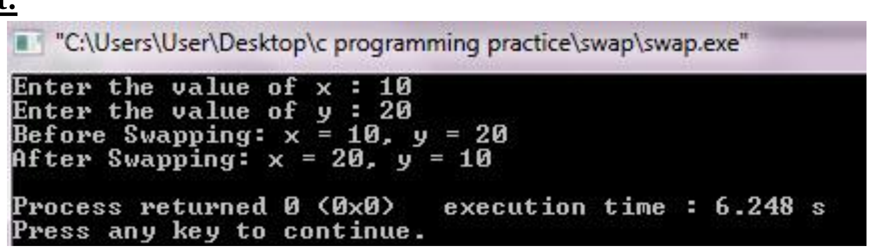
//function definition
void swap(){
   int x,y;

   printf("Enter the value of x : ");
   scanf("%d",&x);

   printf("Enter the value of y : ");
   scanf("%d",&y);

   printf("Before Swapping: x = %d, y = %d\n",x,y);
      x = x * y;
      y = x / y;
      x = x / y;
   printf("After Swapping: x = %d, y = %d\n",x,y);
}
```

<u>Output:</u>



b) <u>Source Code:</u>

```
#include<stdio.h>

void swap();//function declaration

int main()
{
   swap();//function call
   return 0;
}

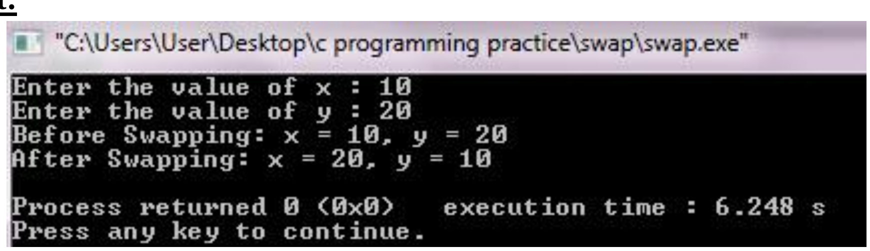
//function definition
void swap(){
   int x,y;

   printf("Enter the value of x : ");
   scanf("%d",&x);

   printf("Enter the value of y : ");
   scanf("%d",&y);

   printf("Before Swapping: x = %d, y = %d\n",x,y);
      x = x ^ y;
      y = x ^ y;
      x = x ^ y;
   printf("After Swapping: x = %d, y = %d\n",x,y);
}
```

<u>Output:</u>



c) <u>Source Code:</u>

```
#include<stdio.h>

void swap();//function declaration

int main()
{
    swap();//function call
    return 0;
}

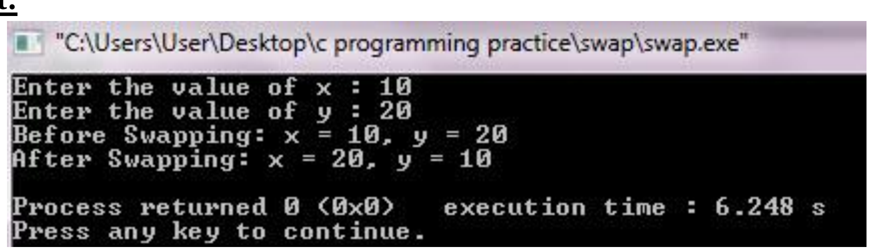
//function definition
void swap(){
    int x,y;

    printf("Enter the value of x : ");
    scanf("%d",&x);

    printf("Enter the value of y : ");
    scanf("%d",&y);

    printf("Before Swapping: x = %d, y = %d\n",x,y);
        x = x + y;
        y = x - y;
        x = x - y;
    printf("After Swapping: x = %d, y = %d\n",x,y);
}
```

<u>Output:</u>



7. Create two functions named square() and area() to perform the following tasks:

- **square()** : to find the square of a number
- **area()** : to find the area of a circle

```
#include<stdio.h>

void square();
void area();

int main(){
    square();
    area();
}

//This segment help us to find the square of a number
void square(){
    int k, s;
    printf("Enter a number : ");
    scanf("%d", &k);
    s = k*k;
    printf("Square = %d\n",s);
}


//This segment help us to find the area of a circle
void area(){
    const float pi = 3.1416,r,k;
    printf("Enter radius of a circle : ");
    scanf("%f",&r);
    k = pi*r*r;
    printf("Area = %f\n",k);
}
```